

MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. CV 10-03561 WHA

**ORACLE AMERICA, INC.'S
OPPOSITION TO GOOGLE INC.'S
SECOND MOTION TO STRIKE
PORTIONS OF THE MITCHELL
PATENT REPORT**

Date: October 13, 2011
Time: 8:00 a.m.
Dept.: Courtroom 8, 19th Floor
Judge: Honorable William H. Alsup

1 **I. INTRODUCTION**

2 Oracle opposes Google’s second motion to strike portions of the Mitchell Opening Patent
3 Infringement Report (“Report”) and the Mitchell Reply Patent Infringement Report (“Reply
4 Report”).

5 **II. ARGUMENT**

6 In the Court’s September 26, 2011 Order, the Court held: “Rule 3-1(c) required
7 identification of the precise element of any accused product that was alleged to practice a
8 particular claim limitation; it did *not* require identification of every evidentiary *item of proof*
9 showing that the accused element did in fact practice the limitation. Google confuses this
10 distinction.” (ECF No. 464 at 4.) In Google’s four new critiques, Google continues to confuse
11 the distinction. As shown below, the precise element of Android that Prof. Mitchell identifies as
12 practicing a given claim limitation is the same in Oracle’s ICs.

13 **A. Response to Critique D: The Report on Infringement of the ’720**
14 **Patent Is Supported by the ICs**

15 Oracle’s ’720 patent infringement theory focuses on a function present in Android called
16 `do_fork()`. Page 19 of Oracle’s claim chart for the ’720 patent identifies Android’s `do_fork()`
17 function as practicing the “copy-on-write process cloning mechanism” element of Claim 1 and
18 quotes from the relevant source code that implements it (`fork.c`). (Declaration of Brian C. Banner
19 (“Banner Decl.”) Ex. C, Oracle ICs Ex. G at 19.) Page 38 of Oracle’s claim chart likewise
20 identifies Android’s `do_fork()` function as practicing the “process cloning mechanism” element of
21 Claim 6, and includes the same source code quote. (*Id.* at 38.) Oracle ICs thus alleged that the
22 `do_fork()` function in Android satisfies both the “copy-on-write process cloning mechanism” of
23 Claim 1 and the “process cloning mechanism” of Claim 6.

24 Prof. Mitchell presents the same `do_fork()` infringement theory in his Report. Prof.
25 Mitchell’s analysis of how Android includes the copy-on-write process cloning mechanism of
26 Claim 1 concludes with an identification of `do_fork()`:
27
28

1 **The file linux-2.6\kernel\fork.c provides the *do_fork()* code**, that calls the
2 *copy_process()* module, **to perform the copy-on-write process cloning** “to
3 instantiate the child runtime system process by copying references to the memory
4 space of the master runtime system process into a separate memory space for the
5 child runtime system process, and to defer copying of the memory space of the
6 master runtime system process until the child runtime system process needs to
7 modify the referenced memory space of the master runtime system process.”

8 (Declaration of Marc Peters (“Peters Decl.”) Ex. A, Report ¶ 614 (emphasis added).) Similarly,
9 Prof. Mitchell’s infringement theory for Claim 6 rests on *do_fork()*. His analysis of how Android
10 includes the process cloning mechanism of Claim 6 concludes with an identification of *do_fork()*:

11 **The file linux-2.6\kernel\fork.c provides the fork code *do_fork()* to perform**
12 **the process cloning** “to instantiate the child runtime system process by copying
13 the memory space of the master runtime system process into a separate memory
14 space for the child runtime system process.” The unset CLONE_VM flag from the
15 *clone()* system call can be passed as an argument for parameter *clone_flags* to
16 *do_fork()*, which passes it to *copy_process()*, which performs the process cloning.
17 As stated previously, because *clone()* does not set the CLONE_VM flag, ***do_fork()***
18 **provides the “process cloning mechanism” of claim 6** “to instantiate the child
19 runtime system process by copying the memory space of the master runtime
20 system process into a separate memory space for the child runtime system
21 process.”

22 (Banner Decl. Ex. A, Report ¶ 627 (emphasis added).) This is one of the paragraphs that Google
23 moves to strike. But Prof. Mitchell’s *do_fork()* infringement theory matches the one in the ICs
24 exactly. In addition, his discussion of the *clone_flags* parameter of the *do_fork()* function,
25 which Google also seeks to strike, is not new either. Oracle identified the *clone_flags*
26 parameter in the ICs for both Claims 1 and 6. (Banner Decl. Ex. C, Oracle ICs Ex. G at 19, 38.)

27 Google’s complaint about Prof. Mitchell’s references to the *fork()* and other system calls
28 mischaracterizes Prof. Mitchell’s reliance on them. He invoked the *fork()* system call, for
29 example, to explain how Android’s Zygote functionality uses *fork()* when it receives a command
30 to fork a new VM instance, and this in turn results in a call to *do_fork()*. (See Peters Decl. Ex. A,
31 Report ¶¶ 598-614.) Similarly, he explained how the *do_fork()* routine provides the different
32 types of process cloning in Claims 1 and 6 by referring to a Linux textbook, and explained how
33 the *fork()*, *clone()*, and *vfork()* system calls invoke *do_fork()* with different *clone_flags* settings,
34 including CLONE_VM (which is unset for both *fork()* and *clone()*). (See *id.* at ¶ 606, pages 302-
35 305.) But this is not a new infringement theory—it is rather further flesh on the bones of the

1 original *do_fork()* theory. To borrow the Court’s metaphor, the discussion of the *fork()*, *clone()*,
2 and *vfork()* system calls is just to explain in which toolboxes the accused *do_fork()* hammer may
3 be found. Prof. Mitchell is permitted to explain in his Report the infringement theories disclosed
4 in Oracle’s ICs. *Sicurelli v. Jeneric/Pentron Inc.*, No. 03-CV-4934 (SLT) (KAM), 2005 U.S.
5 Dist. LEXIS 42233, at *33 (E.D.N.Y. May 3, 2005) (“[E]xpert reports are expected to provide
6 additional information regarding the specific factual bases for plaintiffs’ infringement
7 contentions.”).

8 The ICs gave Google sufficient notice of Oracle’s *do_fork()* infringement theory. The
9 Report adheres to that theory and provides an explanation of the theory, with citations to
10 appropriate evidence. The Court should reject Google’s Critique D.

11 **B. Response to Critique E: There Is No New “mBS Mobile” Infringement**
12 **Theory for the ’476 Patent**

13 Oracle’s ICs allege that Android’s implementation of the Java Security Framework, which
14 includes such Java-API-specified classes as AccessController, ProtectionDomain, and
15 SecurityManager, infringes the ’476 patent. (In fact, these are some of the Java APIs that Google
16 copied from Oracle.) Google moves to strike two identical paragraphs (70 and 77) of the Report
17 that discuss an Android application called “mBS Mobile.” But neither the ICs nor Prof. Mitchell
18 accuse the “mBS Mobile” Android application of infringement. Instead, Prof. Mitchell discusses
19 the mBS Mobile application to describe the benefits that application developers receive from
20 Google having included the infringing code within Android. The paragraphs that Google moves
21 to strike begin: “Some of the ways that application developers could benefit from the
22 java.security framework are illustrated by descriptions of the mBS Mobile application.” (Banner
23 Decl. Ex. A, Report ¶¶ 70, 77.) These paragraphs, by their very terms, do not disclose an
24 infringement theory; they describe a “benefit” of infringement.

25 The Report discusses the mBS Mobile application in connection with the benefit offered
26 by Android’s infringing functionality to rebut Google’s contentions on two points. First, the mBS
27 Mobile evidence establishes that the infringing code that Google made was used, which rebuts
28 Google’s contention that the code was never used. Second, it establishes that SecurityManager

1 was enabled in Android—in fact, it was functional in at least all versions of Android through
2 version 2.2. The Court held that Patent Local Rule 3-1(c) does not require identification of every
3 evidentiary item of proof of infringement. (ECF No. 464 at 4.) Prof. Mitchell’s discussion of
4 mBS Mobile falls into this category. The Court should reject Google’s Critique E.

5 **C. Response to Critique F: The Reply Report on Infringement of Claim**
6 **14 of the ’476 Patent Is Supported by the ICs**

7 Oracle objects that Google did not obtain leave to move to strike any part of the Reply
8 Report, as this is beyond the scope of its August 30, 2011 précis and the Court’s September 26,
9 2011 Order, which concerned only the Report. For this reason alone, the Court should deny
10 Google’s Critique F.

11 Oracle’s ICs identify Android’s AccessController class as containing instructions that
12 perform Claim 14’s “determining whether said action is authorized based on an association
13 between permissions and a plurality of routines in a calling hierarchy associated with said
14 principal.” (Banner Decl. Ex. D, Oracle ICs Ex. E at 22-23.) The ICs disclose that
15 AccessController does the determining by calling the checkPermission method of Android’s
16 AccessControlContext class. (*Id.* at 23-24.) The ICs quote from Android’s AccessController
17 source code to specifically identify that its checkPermission method is declared to be a “public”
18 and “static” method. (*Id.* at 23.)

19 Google’s complaint is that Prof. Mitchell’s Reply Report addresses infringement when
20 SecurityManager is disabled. But there is nothing in the infringement theory in Oracle’s ICs that
21 *requires* Android’s SecurityManager class for Android to infringe. It is true that use of Android’s
22 SecurityManager results in infringement, but that is because SecurityManager calls
23 AccessController’s checkPermission method. This fact was disclosed in the ICs (*id.* at 21) and
24 was discussed in Prof. Mitchell’s Report. (Banner Decl. Ex. A, Report ¶ 736.) But there is
25 nothing in either Oracle’s ICs or the Report that limits the infringement theory to
26 SecurityManager alone.

27 The paragraph of Prof. Mitchell’s Reply Report that Google moves to strike rests squarely
28 on the identification of AccessController in Oracle’s ICs. Here it is in full:

1 Even if the SecurityManager is disabled, the Java security framework may still be
2 used. For example, “the static methods in AccessController are always available to
3 be called.” (L. Gong et al., Inside Java 2 Platform Security 109 (2nd Ed. 2003).)
4 Even if no SecurityManager is instantiated, Android code includes
AccessController.java and the static methods associated with the class may be
called. Therefore, the functionality of the Java security framework is accessible via
the AccessController.

5 (Banner Decl. Ex. B, Reply Report ¶ 115.) In this paragraph, Prof. Mitchell explains, with
6 reference to a textbook written by the ’476 inventor, that Android includes AccessController and
7 its static methods whether or not a SecurityManager object was ever created (instantiated). This
8 is a reference to the accused AccessController and its static method checkPermission that was
9 identified on page 23 of Oracle’s ICs. It is not a new infringement theory because it is not
10 pointing to any new accused functionality; it is merely pointing out that infringement occurs even
11 if SecurityManager is disabled.

12 Google’s Critique F is thus directly analogous to the critique regarding DvmDex.h that the
13 Court rejected in Google’s first motion, and should be rejected for the same reason. (ECF
14 No. 464 at 4 (“[T]he Mitchell report identified specific *functions* located within that file as
15 satisfying the storing limitation. Those functions also were identified in the relevant portion[s] of
16 Oracle’s infringement contentions.”).) *DataTreasury Corp. v. Wells Fargo & Co.*, No. 2:06-CV-
17 72 DF, 2010 U.S. Dist. LEXIS 110658, at *23 (E.D. Tex. Sept. 13, 2010) (denying defendant’s
18 motion to strike portions of plaintiff’s infringement expert report because report did “not
19 substantially deviate from [the] infringement contentions”). The Court should reject Google’s
20 Critique F.

21 **D. Response to Critique G: The Reply Report on Infringement of the ’520** 22 **Patent Is Supported by the ICs**

23 Oracle again objects that Google did not obtain leave to move to strike any part of the
24 Reply Report, as this is beyond the scope of its August 30, 2011 précis and the Court’s September
25 26, 2011 Order, which concerned only the Report. For this reason alone, the Court should deny
26 Google’s Critique G.

27 Oracle’s ICs identify the Simulator class (and, in particular, its “simulate” methods) as
28 performing the step of “simulating execution of the byte codes of the clinit method against a

1 memory without executing the byte codes to identify the static initialization of the array by the
2 preloader” that is recited in Claim 1 of the ’520 patent. (Banner Decl. Ex. E, Oracle ICs Ex. F at
3 11-15.) Google described Simulator as the “Class which knows how to simulate the effects of
4 executing bytecode.” (*Id.* at 11.) As the ICs disclose, Simulator employs several helper methods
5 to perform its simulating functions, including the `parseInstruction` and `parseNewarray` methods.
6 (*Id.* at 13, 14, 15) For the “storing . . . an instruction” step of Claim 1 of the ’520 patent, Oracle’s
7 ICs identify the “writeDex” method of the `Main.java` file as performing the step. (*Id.* at 21-22.)

8 The quotation in Google’s brief of Prof. Mitchell’s in paragraph 92 of the Reply Report is
9 incomplete. (Mot. at 5.) The full paragraph that Google moves to excise portions of is:

10 `parseNewarray` is part of the functionality and operation of `Simulator.java`, the
11 “class which knows how to simulate the effects of executing bytecode,” and which
12 Prof. Parr concedes simulates bytecodes. It is not uncommon that functionality be
13 carried out or span multiple classes of files. The `dx` tool was designed to receive
14 class files that had been created by a Java compiler, and `parseNewarray` interprets
15 the array initialization bytecodes that result from compilation to determine the
16 static values, which are then used to create (and store) one or more instructions
17 requesting the static initialization of the array. The claims do not require the
18 simulating step to succeed in identification of the static initialization of the array in
19 all cases, which is why Prof. Parr’s example in his Appendix A, Section II is not
20 evidence of noninfringement. **Actually, it is evidence of infringement, because
the operation of the Simulator on his modified T.class file produced
instructions requesting the static initialization of the array, which were stored
in the output .dex file.** I note that those instructions were fewer than the original
number of bytecodes from the .class file, which is the purpose of the ’520 patent.

21 (Banner Decl. Ex. B, Reply Report ¶ 92 (emphasis added).) Note that Prof. Mitchell did not, as
22 argued by Google, identify the number of instructions stored in the .dex file as the reason for
23 infringement. By use of an ellipsis to merge two different sentences, Google in its brief removed
24 the text in which Prof. Mitchell identified “the operation of the Simulator” as his reason for
25 finding infringement.

26 There is no infringement contention in paragraph 92 of the Reply Report that is not in
27 Oracle’s ICs. The analysis rests on Simulator, which was disclosed in the ICs. The purpose of
28 this paragraph is to *respond* to an argument made by Google’s noninfringement expert Prof. Parr,
who created a class file (the “modified T.class file” referred to above) by manually altering a real
class file with an editing tool and then processing the altered file with the Android `dx` tool. (It is
worth noting here that the non-real-world example created by Prof. Parr was not disclosed in

1 Google's noninfringement interrogatories. (*See* Peters Decl. Ex. B.) When Prof. Mitchell opines
2 that Prof. Parr's example is evidence of infringement rather than noninfringement, it is not a new
3 infringement theory. The "modified T.class" file is not one that is actually produced when
4 compiling Java source code using the real-world Android SDK, it is something Google's expert
5 created. (*See* Peters Decl. Ex. C ("After compiling this class using the standard Java compiler
6 (i.e., into a file named 'T.class'), I was able to use a library called ASM to alter the bytecode
7 within the clinit method of the T.class file.")) Whether or not processing it satisfies one of the
8 '520 patent claim limitations is not something that needed to be disclosed in Oracle's ICs.

9 It would be unfair to permit Google's expert to hypothesize a factual scenario different
10 from real facts and opine that this scenario reveals something about the operation of the Android
11 SDK that is evidence of noninfringement, and yet forbid Oracle's expert from responding to that
12 opinion. All that Prof. Mitchell has done in paragraph 92 of the Reply Report is apply the
13 Simulator-based infringement theory that was disclosed in the ICs to the new, hypothetical factual
14 scenario created by Prof. Parr. Because that is permissible for experts to do, Google's attempt to
15 take a scalpel to paragraph 92 must fail.

16 Another reason that the Court should reject Google's Critique G is because it is directly
17 contrary to the position Google took in its noninfringement interrogatory response. On August 1,
18 2011, Google argued that Android does not infringe any asserted claim because "it would not
19 employ a method that creates or stores a single instruction to perform each of the respective
20 accused functions **in that there are multiple instructions identified in Exhibit F** [Oracle's '520
21 infringement chart]" (Peters Decl. Ex. B at 27 (emphasis added).) As even Google's own
22 interrogatory response acknowledged, the ICs support a more-than-one-instruction infringement
23 theory.

1 **III. CONCLUSION**

2 The Report and the Reply Report are fully supported by the ICs. Prof. Mitchell has
3 adhered to the infringement theories that Oracle disclosed. For the foregoing reasons, Oracle asks
4 this Court to deny Google's second motion to strike.

5 Dated: October 4, 2011

6 MICHAEL A. JACOBS
7 MARC DAVID PETERS
8 DANIEL P. MUINO
9 MORRISON & FOERSTER LLP

10 By: /s/ Michael A. Jacobs

11 *Attorneys for Plaintiff*
12 ORACLE AMERICA, INC.
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28